An environment for content-based image retrieval from large spatial databases

Peggy Agouris *, James Carswell, Anthony Stefanidis

Department of Spatial Information Science and Engineering and the National Center for Geographic Information and Analysis, University of Maine, Boardman Hall no. 348, Orono, ME 04469-5711, USA

Received 30 September 1998; accepted 29 April 1999

Abstract

In this paper, we address the problem of content-based image retrieval using queries on shape and topology. We focus on the particularities of image databases encountered in typical topographic applications and present the development of a spatial data management system that enables such queries. The query requires user-provided sketches of the shape and spatial configuration of the object (or objects) which should appear in the images to be retrieved. The objective of the search is to retrieve images that contain a configuration of objects sufficiently similar to the one specified in the query. Our approach combines the design of an integrated database with the development of a feature library and the necessary matching tools. In this paper, we present our overall scheme, introduce some individual database components, and provide some implementation results. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: image retrieval; feature libraries; shape similarity; topology similarity; matching; spatial databases

1. Introduction

The intelligent retrieval of images from large databases is the focus of substantial research efforts within the computer vision community (Jain, 1992; Gudivada and Raghavan, 1995; Ogle and Stonebraker, 1995; Pickard and Minka, 1995; Cohen and Guibas, 1996; Cox et al., 1997; Gupta et al., 1998). The objective is to retrieve specific images from a large database by querying the properties of these images. As a result of pioneering research efforts, certain prototype systems have been reported, with some notable examples being Virage (Gupta, 1995), Chabot (Ogle and Stonebraker, 1995), IBM’s QBIC (Flickner et al., 1995), VisualSeek (Smith and Chang, 1995), ImageRover (Sclaroff et al., 1997) and PicHunter (Cox et al., 1997). Most of these efforts address the problem within the context of multimedia applications and therefore, focus on general-use, multimedia-type image databases. In such applications, low-level image properties (e.g., colour and dominant pattern) are often adequate for information retrieval since the image members of the database display substantial differences in these properties and can be distinguished by them alone.

Advancements in remote sensor technology have resulted in increasingly larger and more complex image databases depicting characteristics of the earth’s surface (e.g., topography). Thus, the common practice of using general metadata structures to ac-
cess specific images is becoming less effective and intelligent image query techniques are needed. Topographic image databases, however, are unique in the sense that they contain large numbers of images (typically aerial and/or satellite) which are very similar in terms of general low-level image properties. General-purpose image retrieval approaches such as those mentioned above are not as effective for information retrieval in topographic applications. What distinguishes images in a topographic database is their actual content: the shape and configuration of the objects they contain. Accordingly, shape and topology are more suitable primitives for use in topographic image retrieval systems.

Common approaches to shape-based retrieval systems often employ complex shape descriptors to establish indexing structures. Using these descriptors, shape similarity becomes a nearest neighbour-type search in multidimensional spaces, often supported by tree structures like k-d trees (Stein and Medioni, 1992a,b; Califano and Mohan, 1994; Del Bimbo et al., 1994; Beis and Lowe, 1997). While these techniques tend to be reasonably successful, a common pitfall is the relatively low interpretation potential of the retrieval failures. The similarity of a specific shape to another is often hidden behind complex descriptor metrics.

In this paper, we address the development of a more direct and organised approach that uses raster shapes in support of shape-based image queries. In Section 2, we present system architecture considerations behind Image Query (IQ), our prototype system for image retrieval. We provide an overview of a matching tool to support such queries (Section 3) and emphasise the role and organisation of feature libraries for image retrieval (Section 4). In Section 5, we address the use of topology in queries. Some experimental results and our future research plans are presented in Section 6. It should be mentioned that while our research originates from topographic applications, the developed methodology could be applied to any type of imagery.

2. System architecture issues

In our view, a searchable topographic database comprises images, outline/object information for physical entities depicted in these images and metadata. Metadata of interest include the typical information that describes and enhances the content and/or properties of common topographic data files (e.g., sensor characteristics, date of capture and scale). A schematic description of the operation cycle in our system is shown in Fig. 1.

Our goal is to use as query input: (1) the outlines (i.e., a sketch \(^1\)) of an object or a configuration of objects; and (2) metadata. The operator provides this input information and the database is searched to yield images which satisfy the given metadata parameters and in which appear objects (and, if applicable, spatial configurations) similar to the given sketch. This allows us to take advantage of the intuitive method humans use to represent or express spatial scenes, namely sketching. In order to support such query scenario, our searchable database com-

---

\(^1\) The term sketch is used here in a broad sense. It can refer to an on-screen drawing of an outline using software tools. It can also refer to a pre-existing outline (e.g., from a digital image or a scanned map) selected by the user.
prises three components: image, metadata and feature libraries (Fig. 2).

- **Image library:** contains one entry for every image contained in the database and provides a link/pointer to the corresponding filename.

- **Metadata library:** contains a listing of potential values for a set of attributes that describes general properties of the images and links to members of the image library. These attributes include date and time of acquisition, date and time of introduction in the database, scale/resolution and location of the image expressed in hierarchically arranged geographic entities such as state, county and city. For more complex databases, the attributes may be extended to incorporate information on the sensor and imagery type (e.g., B/W, colour or hyperspectral).

- **Feature library:** is an organised arrangement of distinct feature outlines (i.e., object shapes) and links to image files where such features appear. The role of the feature library is to provide the crucial link that allows us to reduce the search space of a query from a large image database to an abridged group of features.

In addition to these three libraries, a semantic library may exist that contains semantic object information such as use and ownership. Using the above database design during the on-line part of a query, the input query outlines are matched to elements of the feature library using an on-line matching tool. Acceptable matches then give links to specific image files and locations within those images. A comparison to the input metadata values renders certain images invalid candidates. If a semantic library is also used, the matching results will have to pass through another check whereby the semantic properties of the detected objects will be examined. The query results are returned to the user who then has the option to edit his/her query.

In order to support the above process, another sequence of actions has to be performed off-line every time an image is introduced into the database. The user manually inputs the appropriate metadata information for this image and the metadata library is updated to add the new entry. Subsequently, objects/features are extracted from the input image using any digital image analysis tool and these new features are compared to the existing feature library using our off-line matching tool. Library entries are updated accordingly to include links to the objects existing within the newly introduced image. Links between metadata and feature libraries are also updated to connect the metadata values of the new image to the detected features.

The rationale behind our database design becomes apparent when analysing the meaning of the metadata, the feature libraries and their connection. Assuming that we have $n$ distinct metadata values, a point within the $n$-dimensional metadata space corresponds to all images of the same area, captured at the same scale, at the same date and with a similar sensor. When one or more of these parameters can accept less specific values, we move from points to ‘blobs’ within the $n$-dimensional metadata space. For example, photos of various scales of a specific area taken on a specific date form a blob in the metadata space. This blob represents the scale space.
of the area at the time of data capture. By defining points (or blobs) of the metadata space, we actually define a set of representations of a specific geographic area and of the features within it.

3. Matching tool

We have developed a matching tool to handle both the off-line (population and update of the feature library) and on-line (comparison of query input to the feature library entries) parts of this project. Conceptually, our matching algorithm is a variation of least squares matching (LSM), modified to function with edge files. It is well-known that LSM offers a robust method for establishing correspondences among image windows. Its mathematical foundation, based on least-squares principles, permits its successful extension for application in multiple image matching (Agouris and Schenk, 1996), feature-wise global solutions (Gruen et al., 1995) or even to establish correspondences in sets of 3-D images (Maas et al., 1994).

Our matching tool is based on the analysis of dissimilarities between a template (user-provided sketch) and an outline file window. When a template is compared to an edge file, the template is divided into four quadrants. Each quadrant is matched separately to a corresponding image area, and template pixels vote to remain in place (or move) according to their similarity (or dissimilarity) to corresponding file pixels. Moves can be performed in the ±x and ±y directions, in one of five options: left, right, up, down or remain in place. This resembles the comparison of grey values in LSM and the use of image gradients to identify shifts, rotations and scaling. The sum of the votes within each quadrant is recorded, together with recommendations for the direction and magnitude of move. An example of template repositioning situations may be found in Fig. 3. An analysis of the votes of all quadrants allows us to solve sequentially for shift, rotation and scale parameters in order to reposition the template and better match the file object. The final solution is obtained after a set of iterations. Furthermore, by analysing voting patterns within different quadrants, quadrants may be turned ‘off’ to accommodate occlusions.

During the on-line matching process, a feature sketch template is matched against the feature library elements using the same matching tool. Once the template has settled onto a match, the matching accuracy is determined by how many of the template pixels continues to vote to move compared to those that vote to stay put. The comparison of a template to various feature library entries will result in matching accuracy estimates. These matching estimates can be used to determine ranking of different matches. During the off-line matching process, feature library templates are compared to outline files (e.g., edge files or GIS layers co-referenced to digital image files) to establish matches and links to the corresponding image files.

4. Feature library: hierarchy, structure and maintenance

The feature library permits us to narrow the search space from a large database of images to a limited group of feature outlines. In order for the query to be efficient, the library needs to be organised in an optimal way. The ideal optimality criteria are two: the members of the library should be exhaustive (thus being able to describe all possible query input features), and the members of the library should be independent (minimising unnecessary duplications). The two properties, when satisfied, are equivalent to
an ideal library, which is approaching a base spanning the space of shapes.

We have developed a feature library that uses a hierarchical (tree-like) structure (Fig. 4) to approximate the abovementioned ideal optimality criteria. The organisation of data within the feature library is intended to enable it to act as a progressive filtering mechanism. Matching results at a specific tree level are used to eliminate complete branches and, alternatively, to consider only few branches. This reduces the search space from the complete tree to a progressively refined limited subset of branches. The library organisation employs two parameters: a matching percentage above which objects are considered to be same, and a percentage range within which objects are assumed to be similar. The similarity range is obviously upper bound by the lower rate of the sameness range. Combined, the two parameters define the degree to which the feature library approaches the ideal library outlined previously. High values for same ensure the independence of elements at the same level of the library tree. In Fig. 4, the 80–100% range was used to define same and the 50–79% range to define similar. During a query loop, a query sketch is first matched against the features at the parent level in the tree. The process then progresses to child, grandchild and other levels as necessary. Using these intervals for illustration, the hierarchical levels of the library are organised in the following manner.

• The primary parent level: The primary parent level is the level against which every new query shape is first tested and its respective matching percentage is recorded. Features at this level are the roots of the feature library. Different features within the parent level are considered dissimilar if their mutual similarity percentages are lower than 50%. When a query is performed, the matching percentage between the query feature and a parent feature will fall in one of three ranges: 0–49, 50–79 and 80–100%. In the latter case, with a matching percentage in the 80–100% range, the query feature is considered to be the same as the parent feature. The links of the parent feature to files in the image database are then returned as the results to the query. If more than one parent feature matching 80 to 100% to the query feature exists, the links of the top matching parent (higher matching percentage) are returned first, with the links of the remaining matches following. In practice, this case of multiple parent matches is unlikely due to the mutual dissimilarity of the parent features. If the matching percentage is in the 0–49% range, the query feature is considered to be ‘different’ than the parent feature. If this holds for all parent level features, the query feature is a candidate to be inserted into the feature library at the primary parent level. Subsequent off-line processing is then required to establish its links to images in the database.

• The child level: If the query feature matched in the 50–79% range to any parent feature, it is considered ‘similar’ to the parent and is then tested against this parent’s respective child features. In the case of multiple parent candidates, we select the one with the highest percentage first and continue with other candidates in order. If there are as yet no child features for the selected parent, the query feature gets inserted into the feature library as a descendent of the best-matched parent. Its links are then added and prioritised through off-line matching. If child features already reside at this level for any of the selected parents, the query feature is matched against each of them. For a matching percentage in the 80–100% range, the query feature is considered the same as the corresponding child feature and the links of this child feature to the image database are returned as the results to the query. If the matching percentage is in the 0–49% range, the query feature is inserted into the feature library at the child level of the best-matched parent and additional links are added through off-line matching.

![Fig. 4. Example of the feature library hierarchy.](image-url)
The grandchild level: If after matching in the 50–79% range to a parent feature, the query feature matched in the 50–79% range to any child feature, the query feature is tested against the respective grandchild features. Obviously, the relationship between child and grandchild follows that of parent and child. The processes already described take place at this library level to identify the ‘same’ grandchild, to establish a new grandchild or to move further down the tree to the level of great-grandchildren. Combined, these similarity tests have to be satisfied within branches of the tree structure form the insertion-testing criterion. The children levels may be alternatively referred to as n-child levels, with the child being 1-level, grandchild being 2-level, etc.

The matching parameters mentioned here are intended to serve as example values and are not unique. They can be chosen experimentally or even arbitrarily. The percentages may become lower or higher and can be considered as tuning parameters of the approach, with their variations affecting the structure of the feature library in a predictable and organised manner. By narrowing the high (same) matching range — e.g., using the range 90–100% instead of 80–100% — we increase the number of distinct entries at each level of the tree structure, making the tree wider and more shallow. This will increase the search space and make the processing time longer. However, it would also allow the queries to become very specific. For example, the queries might be: ‘retrieve images containing features which look exactly like the query sketch’, as opposed to ‘retrieve images containing features that look like the query sketch’. Widening the high range — e.g., using a 70–100% matching range — will result in fewer entries in the database and will make the tree deeper and narrower. In turn, this produces faster queries with less accurate results. Narrowing or widening the second (similar) matching range will have comparable effects to the structure of the feature library.

The above-described method, whereby new features are compared to existing ones and progress through tree levels until they are eventually matched to an entry or inserted in the library, might result in some inconsistencies. It is possible, for instance, that two sub-children from different primary parents match closer together than to any other feature currently within their respective tree hierarchies. This and other potential inconsistencies are rectified by applying general feature ‘housecleaning’. During feature housecleaning, we consider the sequence in which features are inserted in the library to examine the presence of potential inconsistencies. The process is performed off-line at regular intervals, to ensure the continuous proper organisation. The objective is to ensure that all n-child features are residing under their proper parent, and to minimise the theoretical possibility that unnecessary duplicate features could exist somewhere in the tree hierarchy (i.e., different n-child features that are 80 to 100% similar but reside under different tree branches).

Feature housecleaning utilises a self-organising table called the ‘Temporal Feature Index’ (TFI). The TFI is a two-dimensional cross-referencing table of feature filenames together with their respective matching percentages. Features inserted into the feature library are simply appended to this table in the order that they are introduced in the library and all their matching percentages recorded as they occur. Thus, ordering within TFI reflects relative insertion time. Feature housecleaning searches the TFI top-down and compares features to other features inserted after them to detect inconsistencies. A detailed analysis of feature housecleaning operations may be found in Agouris et al. (1999).

5. Handling topology

To extend the queries in configurations of objects, we employ the well-known concept of nine-intersection, describing the major topological relations between areal, linear and point features (Egenhofer and Franzosa, 1995). According to this model, the topological relationships between two objects are expressed by a 3 × 3 matrix whose elements express whether the mutual relationships between the interior, exterior and outlines of two features are empty or non-empty sets.

In order to consider topological relations between objects, we break the query for a spatial scene into two tasks. First, we identify where individual objects, similar to the query outlines, exist in the database. In order to collect this information, we perform a separate query for each of the distinct outlines which comprise the query scene. This results
in lists of links to image locations (one list for each query object). The location of an outline in an image is expressed by the image coordinates of its centroid, as well as the top-left and bottom-right coordinates of the feature’s minimum bounding rectangle (MBR). The topological relations between combinations from these lists are then determined by using the MBR coordinate information. By comparing viable combinations to the topological configuration determined in the query input, we decide whether the configuration constraint is satisfied. For example, when attempting to retrieve raster files that depict a hexagonal building and a cross-like structure separated by an L-shaped object, we identify the locations in the database where each of the three objects exists. Subsequently, we analyse this information to produce the combinations that satisfy the desired configuration. This formulation of scene queries as a combination of a space query and a topological check within our system permits easy and fast query edits. When attempting to edit a query by re-configuring a set of given objects, we do so by re-examining their topological relations only, an operation simpler than re-searching the shape library.

The use of MBRs instead of the actual outlines is dictated by practical considerations. Establishing topological relations between complex vector outlines is a cumbersome and computationally intensive process that is highly unsuitable for query operations. MBR coordinates are determined easily in a straightforward manner by using matching parameters (rotations, shifts and scales) to position a feature library template onto an image coordinate system. The use of MBRs speeds up the topological building process and thus allows for topology to be built and queried in real-time. If we are interested in considering topological relations which extend beyond single image files, we have to substitute image coordinates by ground coordinates. This could lead to the retrieval of sets of images, instead of a single image file, in response to a single query.

Even though it is beyond the scope of this paper, we should mention that topological queries could be extended to include direction and distance measures. However, the reason we first consider topology is because it has been shown that when users interactively provide sketches of object configurations for a query, their topological relations are of foremost importance and of higher reliability (Bruns and Egenhofer, 1996).

6. Experiments and comments

A prototype system of IQ, the image query environment described in this paper, has been implemented on a Windows NT workstation. This system includes a sample database of 188 features that provides links to a database of approximately 70 image files. These features comprise a grouping of objects which were either manually sketched or extracted from the processed image files. Typical examples of manually sketched features were mostly generic shapes (e.g., circular, representing cooling towers; rectangular, representing buildings). Features extracted from the image files, on the other hand, were rather unique in nature (e.g., outlines of airplanes, buildings and other visible image objects).

As a general rule, generic shapes have links to numerous locations and often multiple locations within a single image. Unique shapes, however, have fewer links that tend to be spatially localised. For instance, numerous airplanes may appear in an airport area but none in other areas.

Search and retrieval times for this feature library were tested using a single 180 MHz Pentium workstation. First, we tested the matching algorithm by comparing one feature against a complete but unstructured feature library that was not organised into a hierarchical tree structure like the one described in Section 4 of this paper. In this case, the feature had to be compared sequentially against all features in the library before determining the best match and returning its links. For the abovementioned library sizes, this corresponded to search times averaging around 6 min 30 s per database. On average, it took approximately 2 s to match the query feature against a template in the feature library. Variations in this time interval corresponded to the differences in object complexity, which might require more iterations.

The next step was to organise the features in the feature library into their proper tree hierarchy according to the rules outlined in Section 4. We used the 80–100% range for same, 50–79% for similar, and <50% for different. With these parameters and after house cleaning, the feature library settles in a
tree structure that is 98 features wide and up to four child levels deep (after the parent level), with 23 features removed as unnecessary duplicates. This corresponds to a trimming of search time nearly in half, as the minimum number of features to compare against a query template is reduced from 188 to 98. Considering that the tree structure is up to five levels deep (parent plus four child levels), the time cost (at 2 s per feature matching) to move down a tree branch is rather minimal. It should be noted here that as the feature library increases in size, its momentum also increases. In other words, a 10-fold increase in the number of templates would not correspond to similar increases in the size of the feature library tree structure. Instead, the number of duplications (within the same range) would increase substantially with few new templates added in the tree structure itself.

We then proceeded with examining the effect on the feature library structure of changes in the same and similar intervals. Table 1 shows the change in the percentage of features included in the feature library as the lower bound of the ‘same’ percentage changes between 100 and 51%. During these experiments, the ‘similar’ percentage was kept between 50% and the lower bound of ‘same’. The table shows, e.g., that assuming the sameness interval to be 75–100% (and correspondingly, the similarity interval being 50–74%) reduces the size of the feature library by 11% compared to when a strict 100% is used to determine sameness.

While Table 1 shows the effect of the sameness interval on the total number of distinct features in the library, Table 2 shows the effects of variations of the similarity interval on the structure of the library tree. Specifically, Table 2 shows how the percentage of parent features changes for varying similarity intervals (for a fixed 80–100% sameness interval). We can see how by broadening the similarity interval, features in the library migrate down the tree branches and the number of parent features is reduced. For example, for a 50–79% similarity interval, we expect approximately 57% of the features to reside at the parent level. By reducing the lower limit of similarity, we reduce the percentage of parent features in the tree. Accordingly, this can be used to reduce the search time. The obvious cost for this would be less precise results.

Table 2 shows an example of the query results. The query element is shown in the upper left window. It matches successfully to an element of the feature library shown in the lower left window. The library element was introduced in a previous operation and was extracted from an actual image.

These early results support the effectiveness of the overall strategy and of the hierarchical organisation of raster feature templates for image queries. Regarding image queries specifically, one should consider that the size of the feature library increases at a much slower rate than the increase of the image library. The feature library eventually reaches a critical mass after which additions of new features are rare, since objects within new imagery are already sufficiently represented in the feature library and there already exist similar templates within it. By transferring the query search space from the image database to the feature library, we manage to limit this search space and increase the effectiveness of our technique.

Our future work plans include the extension of our experimental feature library and image database to include diverse topographic datasets, especially topographic maps. Furthermore, we plan to extend our system to function for spatio-temporal queries. In a manner similar to the approach presented above, spatio-temporal queries can be formed as combinations of spatial queries where two (or more) sets of query results are forced to satisfy certain temporal constraints. For example, we can attempt to retrieve sets of images depicting configurations of buildings before and after the addition of a new wing. This is
Fig. 5. A query template (top left) matches a feature in the library (lower left) and receives its links (marked by windows in the images) as response.

equivalent to searching for the before and the after configurations separately and selecting from the two pools of candidate images those files that depict the same region but differ in their temporal metadata.
Acknowledgements

This work was partially supported by the National Science Foundation through grant no. SBR-8810917 for the National Center for Geographic Information and Analysis, and through Career grant no. IRI-9702233.

References


